This page is left blank

This page is left blank

Author: J. L. Edwards, Artificial Intelligence Management and Development Corporation
Contractor: Artificial Intelligence Management and Development Corporation
206 Keewatin Ave., Toronto, Ontario, Canada M4P1Z8
Contract: W7711-007661/001/TOR
Contract Scientific Authority:
Mr. Keith C. Hendy
Head, Systems Modelling Group
(416) 635-2074
Establishment: Defence and Civil Institute of Environmental Medicine (DCIEM)
Document No.: DCIEM CR 2001-126
Date: April 30, 2001
Report No.: AIM (AC213, April, 2001)

Building an Agent Technology for Users of the *LOCATE* Workspace Layout Tool

- Final Report -

AIM
Artificial Intelligence
Management and Development Corporation

Abstract

Work developed a tight integration between *LOCATE's* ability to track user activity and a new *LOCATE* Agent Network whose ultimate goal is to help ease the data entry burden for *LOCATE* users. A society of agents was created using the JATLite development environment and KQML communication protocols. Names of newly created *LOCATE* objects are sent to an agent network that identifies user needs, filters the search, retrieves human-factors related information and notifies *LOCATE* when the found information is available for display.

An extension to this contract generalised the *LOCATE* work to the area of manning reduction on naval vessels. A Manning Agent Network was created with additional filtering mechanisms beyond those used for the *LOCATE* work. Agents conduct periodic searches of the Internet using the same search and filtering techniques, compare the results against a database of previously found information, identify that is new and add it to the database for display.

Future work on *LOCATE* will use retrieved information to extend *LOCATE's* internal, intelligent-aiding support for data entry. More effective filtering will be added. Future work on the Manning Agent Network will extend the functionality of its filtering agents and refine its search strategies

Résumé

On a procédé à une intégration étroite de la capacité de LOCATE de localiser l'activité de l'utilisateur à un réseau d'agents de LOCATE, en vue principalement d'alléger la pression qu'exerce la saisie des données sur les utilisateurs de LOCATE. Un groupe d'agents a été créé au moyen de l'environnement de développement JATLite et des protocoles de communications KQML. Une prorogation de ce contrat a étendu le travail concernant LOCATE au domaine de réduction des effectifs sur les navires de guerre. Les agents fouillent Internet périodiquement en se servant des mêmes techniques de recherche et de filtrage. Dans les futurs travaux sur LOCATE, on utilisera l'information saisie pour renforcer le soutien interne en aides intelligentes apporté à la saisie de données. Les futurs travaux sur le réseau d'agents viseront à augmenter la fonctionnalité des agents filtreurs et des stratégies de recherche du réseau.

This page intentionally left blank.

Executive Summary

The *LOCATE* Workspace Layout Tool is both a Computer-Aided Design (CAD) environment that allows users to create workspace or facility layout designs and a program for evaluating the communication efficiency of those designs. To use *LOCATE* productively requires an understanding of the nature of its data requirements, an ability to identify and access information relevant to data decisions for a particular design problem, and a skill at using that information to decide which data best characterise a given design problem.

Agent technology provides an opportunity for *LOCATE* and other related software applications to search for, identify and retrieve information and data over the Internet. In the case of *LOCATE*, the results of such search and retrieval could be important in helping users to use *LOCATE* more effectively when addressing its data entry requirements.

LOCATE has the ability to track nearly all of the interface activities of its users when creating a design and to infer the goals they are trying to achieve and the plans they are using to achieve those goals. Work on this contract:

- developed a tight integration with *LOCATE's* ability to track user activity and the development of a *LOCATE* Agent Network;
- created a small society of agents for information retrieval about selected aspects of user activity with the ultimate goal that the use of agent technology help ease the data entry burden for LOCAE users;
- used the JATLite development environment and KQML communication protocols to support the construction of the agent network;

LOCATE and its agent network now co-operate in the following ways·

- when users create and name objects in a *LOCATE* design space, the names of those objects are passed to its agent network;
- the network identifies user needs, filters the search, retrieves human-factors related information on the name and notifies *LOCATE* when information is available for display;
- *LOCATE* displays an alert to let the user know that information has been found for that object;
- the user clicks on a lightbulb alert attached to the interface object for which information has been retrieved and is taken to a Java Window that displays that information. Results are obtained from a search of the Defence and Civil Institute of Environmental Medicine's (DCIEM's) recently implemented Human Factors Engineering (HFE) Guide site and from a more general search of the Internet using MetaCrawler;

An extension to this contract generalised the *LOCATE* work to the area of manning reduction in a naval context. A Manning Agent Network, similar to the one constructed for *LOCATE*, was created. The network:

- includes some additional filtering mechanisms to those developed for *LOCATE*.
- targets four sites for examination:

- the US Department of Commerce's National Technical Information Service (NTIS),
- the US Science and Technology (S & T) Manning Affordability site,
- the US Navy's DD21 site;
- the MetaCrawler site for a more general search of the Internet.

- provides a "Modify Search" window where users may change search items and resubmit a search. The window contains:
    - default search terms;
    - pre-filtering search terms used to exclude items;
    - several post-filtering search options used in more in-depth searching of web pages, abstracts and other documents.
- performs periodic searches, compares the results against a database of information and identifies any new information from a site that is to be added to the database.

Future work on *LOCATE* and Manning Agent Networks will:

- use retrieved information to extend *LOCATE's* own internal intelligent aiding support for data entry;
- perform more effective filtering of *LOCATE* results to help provide useful supplemental information on items for which function and priority weight data must be entered into *LOCATE*.
- extend the functionality of its filtering agents;
- extend and refine the search strategies.

Edwards, J. L.. 2001. Building an Agent Technology for Users of the ***LOCATE*** Workspace Layout Tool. AC213 Defence and Civil Institute of Environmental Medicine.

Table of Contents

List of Figures

Background

Recent work on the *LOCATE* workspace layout tool has added a number of new features and enhanced the maturity of the application. In addition to providing important support in a variety of workspace layout contexts, *LOCATE* now serves as a testbed for developing ideas and techniques for intelligent aiding.

LOCATE is both a Computer-Aided Design (CAD) environment that allows users to create workspace or facility layout designs and a program for evaluating the communication efficiency of those designs (Hendy, 1984; 1989).

LOCATE has been used in a number of practical design applications, including the evaluation of bridge configurations for a Tribal class (DDH280) destroyer of the Canadian Forces and the analysis of proposed ship designs for the US Navy, as part of their Surface Combatant—21 (SC-21) project. Further, goal and plan tracking developed to support *LOCATE's* intelligent aiding has been considered as a possible model in a British study to develop and deploy an interface tracking manager for the fast jet domain. Most recently, *LOCATE* was used in a second application with the US Navy to help in the design of SACC LHD—6 layouts.

Recent work on the intelligence testbed (see Edwards and Hendy, 2000) has focused on creating a general approach to action, goal and plan tracking that can be generalised to other application domains. That work likely will prove useful to a number of projects being developed within DCIEM, including a proposed project for the fast-jet domain within the Canadian Forces

As an application, *LOCATE* is useful in large projects where communication efficiency is of considerable, or even critical importance. One barrier for potential users of *LOCATE* is the somewhat specialised knowledge required for data entry.

To use *LOCATE* productively requires an understanding of the nature of its data requirements, an ability to identify and access information relevant to data decisions for a particular design problem, and a skill at using that information to decide which data best characterise a given design problem.

More specifically, users need to understand how the functions that *LOCATE* provides can be used in describing the communication characteristics of their designs, and how priorities of communication among design elements can be determined and set. Having understood those things, users need to know where they can locate relevant information to the data they must enter for their design problem.

Recent developments in agent technology (see e.g., Weiss, 1999, Wooldridge, 2000) provide an opportunity for *LOCATE* and other related software applications to search for, identify and retrieve information and data over the Internet In the case of *LOCATE*, the results of such search and retrieval would be important to solving much of its data entry problem.

The purpose of this study is to design and implement a supporting technology that will deploy agents over the Internet to identify and examine sites that contain information relevant to the data entry requirements of *LOCATE*. The work will interface that information to *LOCATE's* intelligent aiding system in ways that serve to ease the data entry burden for *LOCATE* users.

Research Approach

The specific purpose of building an agent technology for *LOCATE* is to help solve the knowledge acquisition aspect of the data entry problem faced by *LOCATE* users. This work represents a first step in the design and construction of a multi-agent system to provide help in solving that problem. More generally, its purpose is to continue DCIEM's intelligence testbed efforts by extending the principles and techniques of planning and intelligent aiding, not only useful to *LOCATE* but also to other, related applications.

The environments in which this society of agents will operate are the *LOCATE* and Internet/Intranet environments. On the *LOCATE* side, agents will monitor the data entry activity of a user to determine what types of information might be helpful in addressing the user's needs and when that information should be presented. On the Internet and Intranet side, agents will search for and retrieve information and data from various sites, e.g., human factors sites, that could be useful to data-entry decisions.

Several types of agent will be considered: Task-specific and profiling agents that monitor a user's activity within *LOCATE* and on the Internet, information retrieval agents that search for and retrieve information from the Internet and selected Intranets, planning and scheduling agents that determine when the information retrieval agents are to be deployed and what, in a broad sense, they are to do, summarising and collating agents that accept information from the information retrieval agents and determine what information and data should be presented to a user, and finally, agents that either plan how and when the information is to be presented or that interface with *LOCATE's* intelligent aiding system, which itself makes those decisions

In the long view, this system of agents may be extended to support communication with other software applications in the suite of tools of which *LOCATE* is a part.

As part of the current work, a variety of issues will need to be addressed when designing this collaborating, multi-agent system, including: 1) how agents are to know about one another and their respective abilities; 2) how they are to communicate with one another through what common language; 3) what domain-specific or common ontology are they to use; and, 4) how their activities are to be co-ordinated.

Whereas all of the agents envisioned in this multi-agent system qualify on most counts as agents, the boundaries between what is an agent and just another piece of software remain somewhat fuzzy. In designing and building an agent system, criteria for establishing software agents need to be considered, such as the ability of the agent to be **reactive**, so that it can respond to changes in its environment, **autonomous**, so that it is able to exercise control over its own actions, **goal-oriented**, so that it may evidence purpose, and **communicative**, so that it may interact with other agents and possibly human users.

One criterion often cited in defining agents is **persistence**, that is, the agent software must be a continually active process. Although this criterion may apply only to some agents in this system, e g., information retrieval agents, in principle it could apply to all of them.

Other criteria for agents such as their mobility and flexibility should also be considered but are secondary in this preliminary stage of development.

Some time will be spent exploring sources of information that could be helpful to the decisions users need to make about data that *LOCATE* requires. Linking the user quickly to relevant information sources or identifying, retrieving and presenting that information could save considerable time and effort in applying *LOCATE* to a design problem.

One likely source is the set of human factors standards, e.g., MIL-STD-1472, recently implemented in hypertext under separate contract to DCIEM (W7711-9-7579) and posted on a demonstration site on the Internet (http://www.interlog.com/~mallin/HFE_IntroPage.html). Those standards represent important information for users of many applications including *LOCATE*.

Further to Internet related issues, a custom browser for *LOCATE* was implemented in a previous contract but does not support the extensive functionality of the two mainstream browsers: Netscape and Internet Explorer. An examination will determine how *LOCATE* can be linked to those two Internet browsers for the display of some of the information useful for data entry.

Finally, features will be added to *LOCATE* that will make the data entry process more user friendly and offer help to the user in support of that process.

Study Objectives

The study objectives for this project include.

- to identify materials, projects and products of agent technology relevant to building a system of multi-purpose agents to support the knowledge acquisition aspect of the data entry problem for *LOCATE* users;
- to identify and describe environments in which that multi-agent system will function and to define the agents that those environments will support;
- to design a multi-agent system with special attention to issues of agent identification and description, agent communication, a shared ontology, and agent co-ordination;
- to identify external sources of information helpful to users' data entry decisions, e.g., human factors standards, and show how some agents in this society of agents will access and retrieve information from those sources;
- to review the properties of functions that *LOCATE* provides for describing the communication characteristics of design elements;
- to implement, as a proof-of-concept, one of more agents for information retrieval and at least one agent either to determine how and when the retrieved information is to be presented to the user or to interface to *LOCATE's* intelligent aiding system, which itself will make those determinations;
- to implement ways of presenting function–property information to *LOCATE* users in a manner that permits a quick grasp of the nature of the available functions and their differences;
- to examine and, if possible, implement links between *LOCATE* and the two standard Internet browsers.

Designing and Building an Agent Network for *LOCATE*

As a first step in building an agent network to address the needs of users, it was determined that *LOCATE* should identify sources of motivating information, that is, information that arises from user activity and that would motivate agent behaviour. Although such an approach requires refinement for the more specific purpose of easing the data entry burden, it offers opportunities for satisfying other goals as well, namely, goals involved in the creation of a design as well as its analysis. It may even help identify goals that users (especially novice users) need to pursue but may not be aware they need to pursue.

In general, agents monitor and respond to the activities of a user for the purpose of identifying and displaying information that might be helpful to that user. In the current case, a small society of agents was created to monitor the user's activity, assess his or her current needs, retrieve information related to those needs, filter that information, decide if, when and how to display the information and notify the user when pertinent information is available.

More specifically, *LOCATE* monitors the creation and naming of objects in the design space, initiates searches for information related to those named objects, performs filtering of that information, notifies the user that information is available about the target object and, when appropriate, displays it for him.

In addition to a broad general search of Internet sites, the current implementation includes a more focused search of a recently implemented human factors engineering site, that is, the Human Factors Engineering (HFE) Guide site. Among the items on that site are Army and Air Standards, such as MIL-STD-1472 and MIL-HDBK-46855, as well as standards for the aircrew-machine interface.[1] Currently, HFE Guide is being hosted on the following demonstration site: http://www.interlog.com/~mallin/HFE_IntroPage.html.

The next several sections include brief comments on issues surrounding the notion of agent, a discussion of the issues involved in deciding on an agent support environment for building *LOCATE* agents and, finally, a detailed description of *LOCATE's* agent network.

Brief Comments on the Notion of "Agent"

Whereas all of the agents envisioned in *LOCATE's* multi-agent system qualify on most counts as agents, the boundaries between what is an agent and what is just another piece of software remains somewhat fuzzy.

In designing and building an agent system, criteria for establishing software agents were considered *LOCATE* agents are **reactive**, in that they respond to changes in the environment, specifically to user activity and, more specifically in this implementation, to the creation and naming of objects.

LOCATE agents are **autonomous**, in that they are able to exercise control over their own actions. Currently, most exercise only simple control but as decisions become more complex as to when and how often to initiate searches and how those searches are to be conducted, the autonomy of these agents will increase.

[1] More recent work on this site seeks to incorporate the Java-based version of the Guide into a site for Human Systems Integration (HSI) run by Defence Research and Development Canada (DRDC). This includes both their DND Intranet and public Internet www sites.

Although *LOCATE's* agents are not explicitly **goal-oriented**, it should not be difficult to modify them to exhibit goal-oriented activity, a characteristic already part of *LOCATE's* plan generation capability. Since intelligent aiding in *LOCATE* generates goals and plans for how it will provide help to its users, only modest work will be required to extend the now implemented agent network to reflect similar purpose and method in its activities.

Agents in *LOCATE* are **communicative**. The JATlite development environment (see discussion in the next section) uses KQML, a set of communication protocols that support the exchange of information among agents. That standard uses performatives, derived from speech act theory, such as "tell," "ask," "request," etc., for passing information from one agent to another. Similar standards may be used in future to support more direct communication between *LOCATE's* agent network and its human users.

LOCATE's agents currently do not satisfy the criterion of **persistence**, which is often used in defining an agent, that is, the agent network is not a continually active process. Part of an extension to this contract, however, has added periodic searching, which clearly is a form of persistence.

Network action is now triggered not only by user activity but also by the simple passage of time: after a predetermined interval, the agent network re-initiates one or more searches for information searched for at some earlier time. Naturally, returned results are filtered so that only new information is displayed.

Mobility and **flexibility** are two other aspects of the concept of agent. At some future point, it may be useful to deploy mobile agents over the Internet to make contact and share information with other agents. Increased flexibility will come from adding intelligence to the agent network, which is achieved in part by using the model mentioned above on which intelligent aiding in *LOCATE* is based.

Finally, a brief note is in order on the notion of **ontology**. This term arose in philosophy in the 17th century and refers to systematic accounts of existence or being. In that context, ontology is often distinguished from cosmology, psychology and theology.

The use of the term in computer science is in the context of agents, agent networks or "societies," and knowledge sharing. An ontology is a more or less formal definition of the concepts and relationships that exist for an agent or a community of agents. In practical terms, a common ontology shared by a network of agents, such as the one created for *LOCATE*, is given in Gruber. (1995):

> *"Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner. The agents sharing a vocabulary need not share a knowledge base; each knows things the other does not, and an agent that commits to an ontology is not required to answer all queries that can be formulated in the shared vocabulary."*

Ontology is addressed in these practical terms in comments in the next section on the emerging FIPA standard and KQML

Selecting an Agent Development Environment

In deciding how to construct a collaborating, multi-agent network for *LOCATE*, a variety of issues needed to be addressed, including: 1) how agents are to know about one another and their respective abilities; 2) how they are to communicate with one another through a common language; 3) what domain-specific or common ontology are they to use; and, 4) how their activities are to be co-ordinated.

In support of an agent network for *LOCATE*, development environments and various tools and toolkits were examined A principal type of agent was investigated that was considered particularly useful in building an agent technology for *LOCATE*, i.e., an Information Retrieval Agent (IRA).

Retrieval agents search and retrieve information and serve as information brokers and documents managers. Various initiatives have studied and used IRA's and some of those were examined. Some applications are intelligent IR interfaces, others, mediated searching and brokering, and still others, clustering and categorisation Some products that claim to be retrieval agents include the client-based AT1, BullsEye, Go-Get-It, Got-It, Surfbot, and WebCompass, and the server-based Agentware and InfoMagnet. IRA's that were examined somewhat more closely include:

- DVINA - an agent for monitoring the web, USENET, or email using both statistical and knowledge-based techniques.
- WebMate - a personal Web agent integrating parallel search techniques, relevance feedback, similarity-fetching, offline browsing (both of the above from the Intelligent Software Agents Group at Carnegie Mellon);
- Ahoy and Metacrawler - mediate Internet searches (The Softbots group at the University of Washington);
- Adaptive IR agents (The Nobots group at the Stanford Robotics Laboratory);
- Prototype System IRA (Information Retrieval Agent) - searches for loosely specified articles from a range of document repositories (Voorhees);

Examination of those and other IRA projects gave direction to the work on an agent network for *LOCATE* making it clear that an infrastructure was needed to provide a context for the development of those IR and other types of agent. A review was conducted of several agent building support systems using the following criteria

- Is a license required for its use?;
- Is it a robust and usable product as opposed to a research project?;
- Does it provide support for static vs. mobile agent creation?;
- Does it support the use of Java-based expert systems, e g., JESS; CLIPS?;
- Does it support the creation of information retrieval agents?

Beyond these criteria, it was determined that development tools for the Macintosh are slightly behind those on the PC. Since most of AIM's development work in Java to date has used the Macintosh platform, this was a potential area of concern Further investigation indicated:

- Internet Java applications naturally run under any OS regardless of the development platform;
- the most recent version of Java is 1.2 [more recently referred to as Java 2.0];
- Apple's MRJ supports Java 1.1;
- MRJ now provides support for Java 2.0 in OS X but is not backwardly compatible;
- choosing a support system is somewhat restricted if Java 1.1 is used [more widgets and more customisable look for those widgets in Java 2.0; drag and drop between Java applications is supported in ver. 2.0; some added audio capabilities are provided in version 2.0];
- Support systems are available on the PC that permit the use of Java 2.0;

From the above, it was determined that in the short-term, Java 1.1 does not overly restrict the choice of a support environment for building an agent system. Therefore, a support environment could be chosen that permits development on the Macintosh under Java 1.1, with a view to moving that work to Java 2.0 under Mac OS X at an appropriate future date.

A number of projects reported in the literature are agent systems either developed from scratch or making use of agent building tools and toolkits. Most agent systems are being developed for use within the Internet or Intranets, which is the context in which the *LOCATE* agent system was to be developed. Of course, agent systems are not restricted to the Internet and can be constructed for other environments such as LANs and WANs and for handling large database systems.

The following were considered in making a final decision about which support tool(s) to use in building an agent system for *LOCATE*:

- JATLite - the best tool for several reasons: First, it is Java 1.1 compatible and Mac compatibility is advertised. The source code is freely available. It is compatible with the Java Expert System Shell (JESS), which would be useful in adding intelligence to agents. Interestingly, it has been used in extending the capabilities of AutoCAD, a plus since *LOCATE* is a CAD application. Finally, it has been used for creating information retrieval agents;
- Aglets - focus of this tool is on mobile agents, which may become part of the *LOCATE* agent framework in future but is not a focus of the current work;
- DECAF - relies on both Java 1.1 and 1.2 so that to make complete use of the product requires support for the latter, which is not yet available for the Macintosh;
- Gypsy - focuses on the creation of mobile agents;
- JADE - holds promise for future work but requires Java 1.2, not yet available for Mac;
- JAFMAS - Java 1.1 compatible, open source and freely available. More of a research project, however, than a practical tool and so not robust enough for the initial work on a *LOCATE* agent system;
- JIAC - emphasis is on telecom agents that provide user support across a variety of telecom products; computers, palm pilots, phones and the like No downloadable version could be found to aid in further review;
- Zeus - requires Java 1.2, not yet available on the Mac. No explicit support for Information Retrieval Agents (IRA's).

Two agent communication standards were examined : KQML and FIPA ACL. KQML has been around for some time but the FIPA ACL appears to be the evolving standard for communication among agents. Although KQML was used in the current research, the two standards are close enough that moving from one to the other poses little problem. This issue will be reviewed in future work on this project, as greater reliance on agent communication becomes necessary.

LOCATE's Agent Network

An agent network was constructed to support *LOCATE* users in designing workspace layouts and in entering data to analyse the communication efficiency of those layouts. The following agents form part of the *LOCATE* agent network:

- Internal Monitoring "agent" (IMa);
- Information Needs Agent (INA);
- Front-end Filtering Agent (FFA);
- Information Retrieval Agent (IRA);
- Information Display Agent (IDA);
- Application Notification Agent (ANA);
- Internal Alerting "agent" (IAa).

Monitoring Agents.

In deciding on the kinds of agents that would populate *LOCATE's* agent network, several types were considered: Two types of monitoring agents were considered and implemented. The two types are **task-specific agents** and **profiling agents**. Two task-specific agents were designed and implemented during this contract phase An **Internal Monitoring "agent" (IMa)** and an **Internal Alerting "agent" (IAa)** perform two tasks within *LOCATE.* Neither of these is what might be considered a typical agent since each operates from within the *LOCATE* application, although performing different tasks on behalf of the user. [2]

The first of these agents is part of *LOCATE's* background functionality for tracking the actions, goals and plans of its users. The agent detects the creation and naming of objects and passes those names out of the application to the agent network.

The second instance of a task-specific agent is an alerting agent, which receives notification from the agent network that information is available which might be of use to the *LOCATE* user. That agent then decides when and how to put up an alert that the information is available. Currently, this decision is based solely on whether the user has toggled the Agent Alert Menu Item in *LOCATE's* Help Menu on or off but, in future, it will depend on reasoning about the context of user activities.

[2] The acronyms for these two agents use a small "a" to designate the word, "agent," since, as indicated, they are not considered part of the typical agents that make up an agent network. They do use the same communication protocols, however, in exchanging messages with the network.

The second type of monitoring agent, the profiling agent, might be helpful in that very reasoning process. The job of *LOCATE's* **Information Needs Agent (INA)** is to determine what information is needed by the current user. A simple example of this profiling agent was designed for this study. At the moment, it simply passes names of objects that it receives from *LOCATE's* internal monitoring agent on to a filtering agent. Two simple extensions could be added to this agent, however, that would involve maintaining a list of all named objects to ensure no duplicate searches are performed and maintaining a list of the information already retrieved for the current user and passing that on to the front-end filtering agent for use in filtering that information out of any new searches performed.

A few other points can be made about profiling and task-specific agents. First, some profiling already is done in *LOCATE* when the user is asked to make judgements about his or her proficiency in using *LOCATE* and, separately, in designing workspaces. The user makes these judgements in the "About You" window, which may be displayed at startup or later, using the "About You" Menu Item in *LOCATE's* Help Menu. These and other user attributes could become part of information maintained by one or more profiling agents and used in reasoning about the kinds of information that might be useful and in what contexts.

Further, task-specific and profiling agents could have counterparts in the agent network, whose job it would be to track user activities in surfing the Internet and develop profiles from the user's actions in that environment. The resulting knowledge might be co-ordinated with that maintained by *LOCATE's* internal monitoring and profiling agents.

Filtering Agents.

The job of this type of agent is to determine whether search information has been adequately identified and qualified. When accepting retrieved information, another type of filtering agent may have the job of summarising and collating that information and determining which parts should be presented to a user.

A **Front-end Filtering Agent (FFA)** was designed and implemented for the *LOCATE* Agent Network. Currently, this agent adds a set of qualifying terms to the name of a newly created object and passes that along to the next agent for processing. Future work on this agent will provide for contextual information that determines which qualifying terms are to be added to object names before a search is performed. Further, as other types of search are identified and implemented, based on different types of user activity, the functionality of this agent will be expanded

In addition to the FFA, a **Back-end Filtering Agent (BFA)**, also was designed and implemented. This agent accepts information from *LOCATE's* Information Retrieval Agent (see below) and performs filtering on information retrieved One back-end filtering task was implemented during this first phase of agent work on *LOCATE*: an agent goes to each identified site in the results returned from MetaCrawler (the general Internet search engine used in this study) and determines whether the site is legitimate, that is, determines that there are no "Page Not found" errors. If the site is legitimate, the site address is placed in the results for display to the user. If the site is not valid, the site address is tagged as "invalid" and not entered for display.

Another task for this agent that would be fairly easy to implement would be to have it co-ordinate with the User Needs Agent (UNA) to identify information previously displayed to the current user. All new information received then would be compared against information previously provided to that particular user When duplicates were found they would be discarded, leaving only new information to be passed along.

This feature in fact was added to the Manning Agent Network, which was an extension to this contract and a generalisation of the *LOCATE* Agent Network

Information Retrieval Agents.

Considerable work in this contract phase was done on an **Information Retrieval Agent (IRA)**, which was designed to search for and retrieve information from two sources. The first source is a recently constructed site for DCIEM that contains a variety of human factors standards information. The site is the Human Factors Engineering Guide site, more commonly referred to as "HFE Guide."

LOCATE's IRA searches this site, making use of the Guide's own search facility and returning results for possible display to the user. Other targeted sites like HFE Guide could be searched by *LOCATE's* IRA in future. One source of such sites is a list compiled as part of a set of bookmarks for a custom browser implemented in an earlier version of *LOCATE.*

In addition to the targeted HFE Guide search, the IRA also searches the Internet using MetaCrawler. MetaCrawler is a metasearch engine which originated at the University of Washington. It is licensed by InfoSpace (recently merged with Go2Net under the InfoSpace name) which acquired it from Netbot. Metasearch engines make use of other search engines on Web and search considerably more of the Internet than any one search engine. MetaCrawler is one of the best metasearch engines, others being Ask Jeeves, Google, Inference Find and SuperCrawler. Two other metasearch engines, HuskySearch and Grouper, were developed at the University of Washington but have since been retired.

Generally, information retrieval in *LOCATE's* agent system is triggered by user activity and not by any direct interaction with the user. A future exception to this approach is planned and will occur when filtering "requests" through *LOCATE's* "Smart Help" system In that planned modification, if *LOCATE* determines that help can be given locally, , i.e., from within the application's help system the agent system will not invoked; if it cannot, the "request" is passed to one or more IRA's to perform a search.

Planning and Scheduling Agents.

Currently there are no planning and scheduling agents for *LOCATE.* Those agents determine when IRA's are to be deployed and what, in a broad sense, they are to do. Once the agent network is able to determine what information is new, IRA's can be scheduled for periodic search with old information filtered out of the results that are returned. Further, as the types of motivating user activity are extended, decisions about what to search for and when will require more complex processing. At that point, planning and scheduling agents will become useful for *LOCATE.*

Summarising and Collating Agents.

These agents accept information from information retrieval agents and determine what information and data should be presented to a user. This functionality was addressed in discussing the filtering agents, above. Although some simple operations might place those filtering agents into this category, there are many other types of summarising and collating activity that should be considered. For example, criteria might be specified for use in automatic search of identified web sites. The appearance of select keywords would be one way to summarise those pages. Collating could be based on different sets of keywords or more explicitly on the selected sites searched such as the current HFE Guide and other sites from the list compiled for *LOCATE's* customised Internet browser mentioned earlier.

Notification and Presentation Agents.

These agents will plan how and when retrieved information is to be presented to a user or, alternatively, interface with *LOCATE's* intelligent aiding system, which itself will make those decisions. *LOCATE's* ability to track user activity should not be duplicated by its agent network, however, decisions about when and how information is to be presented to a user may necessarily involve an understanding of what the user is doing and the state of information retrieval. Thus, some co-ordination of user and agent activity in making those decisions is likely.

At the moment, presentation of retrieved information is based on whether identified sites have been tagged as legitimate, whether the user has been notified that information is available and whether he or she has indicated a desire to view the information.

Once information has been retrieved and filtered, the agent network informs *LOCATE* that it is available for viewing. *LOCATE* displays an alert in the form of a small lightbulb inside the object for which information has been retrieved. By clicking on a lightbulb, the user is taken to a Java window containing three panels.

The first panel lists all terms for which information has been retrieved, whether or not the user has examined that information. The next two panels identify search results, first, from DCIEM's recently implemented HFE Guide and, second, from a more general search using MetaCrawler. By clicking on an item in either window, an HTML page is opened to display the result. Search results from the HFE Guide are displayed using its standard interface. Currently, the user is allowed to navigate to different areas of the Guide but that will be prevented in future implementations. Topics will be displayed as non-selectable items that allow the user simply to understand which standard contains the found information and where in that standard the information is located. This decision is predicated on the notion that the agent should deliver relevant information to a user and that the user should not be permitted to wander at will in a an extensive and complex set of standards losing valuable time and perhaps the reason why he was taken there in the first place.

Finally, a set of alternative qualifying terms and phrases are provided for the "topic" being searched. This temporary addition was included more as an aid to the development of the interface than as an option for the user. The assumption underlying *LOCATE's* agent network is that it does the searching and presents useful results to a user. The presentation of those results should not encourage users to become distracted by engaging in broader searches by changing the agent's qualifying terms. Of course, a user may make use of a browser to conduct further searches but that should be not be encouraged in this context. An important goal of the agent network should be to find and present the kind of information that will make further search by the user unnecessary.

The Agent Network Interface

Although there are a variety of activities associated with *LOCATE's* agent network, there are only two interface elements that are seen by *LOCATE* users: notification alerts, which inform the user that information is available, and Java and HTML information displays. Initiating a search is a background activity not associated with any interface display. User activities are monitored by

LOCATE and certain of those trigger communication with its agent network to motivate a search. Recall that current searches are tied to the naming of objects in the workspace.

Once a search has been initiated, information retrieved and filtered and results returned, some way of notifying the user that information is available is needed. To notify a user that information is available, one technique considered was the placement of a lightbulb along the top of *LOCATE's* menu It was determined, however, that a more specific notification was possible, one that communicates not only the availability of information but also the topic of that information.

The lightbulb, which is a standard interface technique for indicating help, was used but, instead of displaying it in the menu bar along the top of the screen, it was attached directly to the object for which information had been retrieved. Since named objects in *LOCATE* include workstations and obstructions and since the display of each has an arrow to indicate its current angle of rotation, it was decided to attach lightbulbs to the origin of that arrow as an indication that help information was now available for that object.

Figure 1a below is an example of the *LOCATE* interface with workstations, workstation obstructions and fixed obstructions. The display shows the use of lightbulbs to indicate that information from the agent network is available for display. In the example figure, information is available for a workstation, a workstation obstruction and a fixed obstruction.

Figure 1b shows the results of a selection of the lightbulb on the Workstation that the user has named, "Officer of the Watch," and Figure 2a shows the Java Window used to display the results of the agent network search.

Notice that although results were returned for the general search, no results were found in a search of the HFE guide site. Figure 2b shows the resulting display when the user selects the second item in the list in the middle window of Figure 2a. The agent network has launched the user's default browser and displayed the associated result. The user is now able to use that browser's search facility to look for instances of the named workstation. In this example, the information seems useful in that it deals with movements (links) of an officer of the watch on the bridge of a destroyer

That result is itself a hyperlink to an illustration of those movements and Figure 2c shows that illustration.

In Figure 3a, results are shown for a search of the name of the workstation obstruction (see Figure 1a), i.e., a console. Results were found from searching both multiple search engines (MetaCrawler) and the HFE Guide. Figure 3b illustrates the effect of selecting one of the items returned from the HFE Guide site. The result is displayed in the HFE Guide's window interface and it's "Find In Text" option (see button at bottom of figure) has been used to locate the item in the content window of the Guide on the right.

The final figure illustrates results from a search for the name of the fixed obstruction in the *LOCATE* design, namely, a ladder. The results look much as they do for the previous search. Figure 4 is a display of one of the items selected from those results. Again, the Guide's "Find in Text" facility has been used to locate instances of the search term in the content window of the Guide.

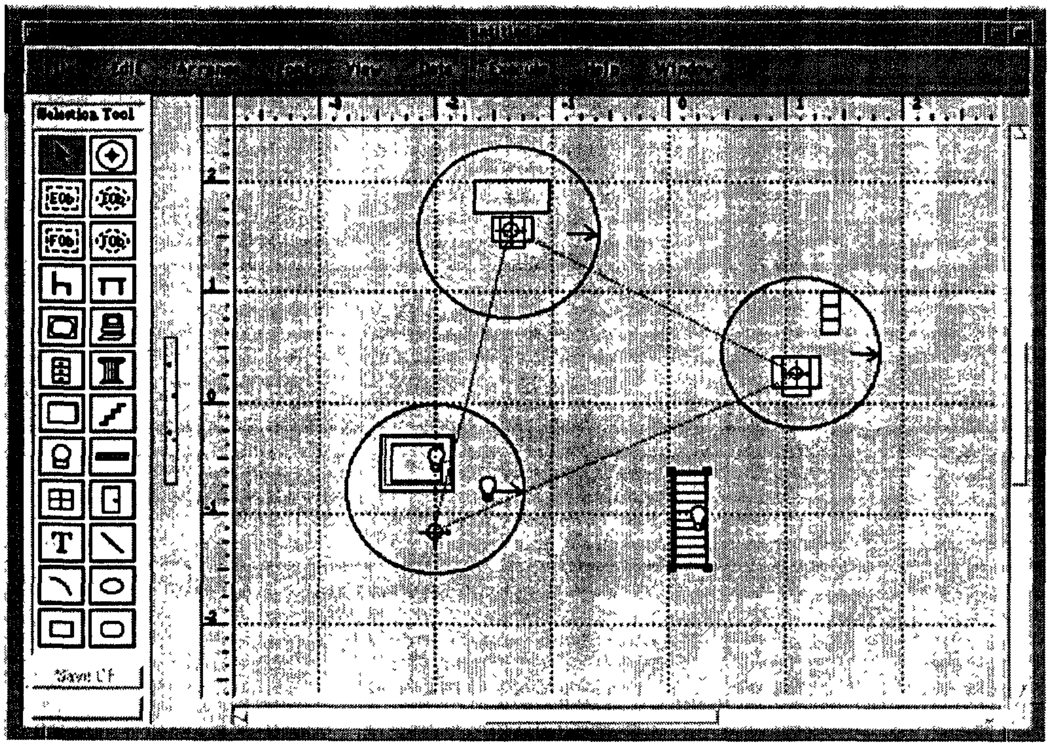


Figure 1a. Lightbulbs Indicating Information Retrieved by LOCATE's Agent Network

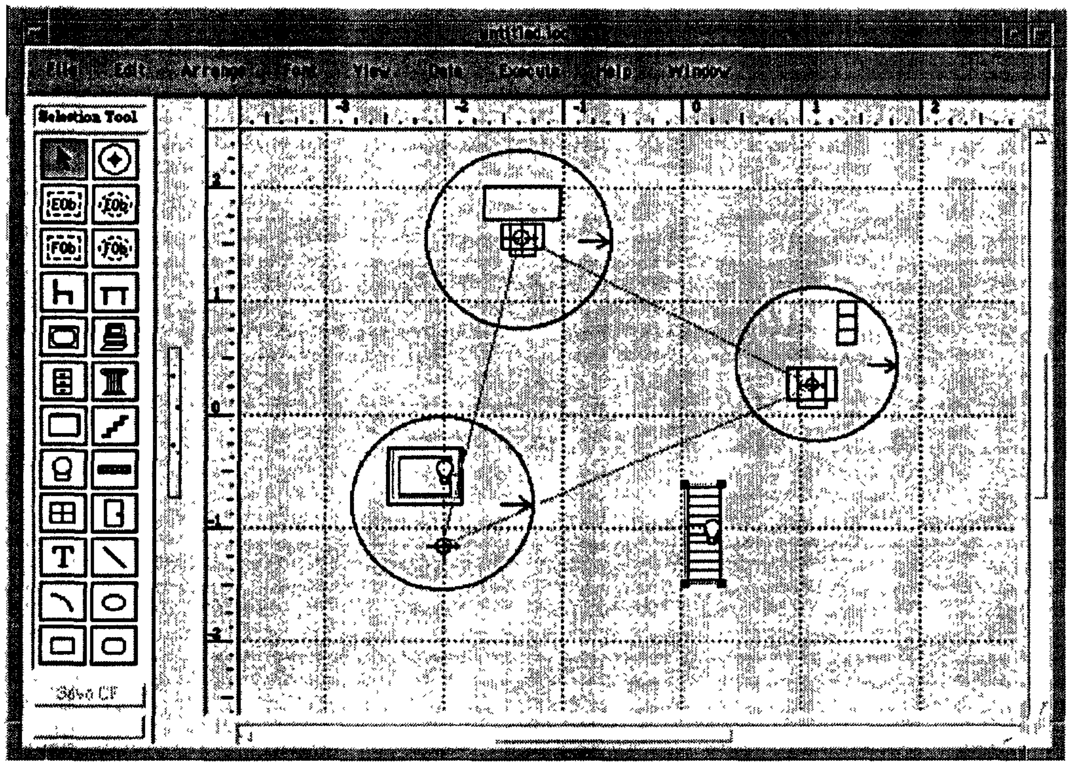
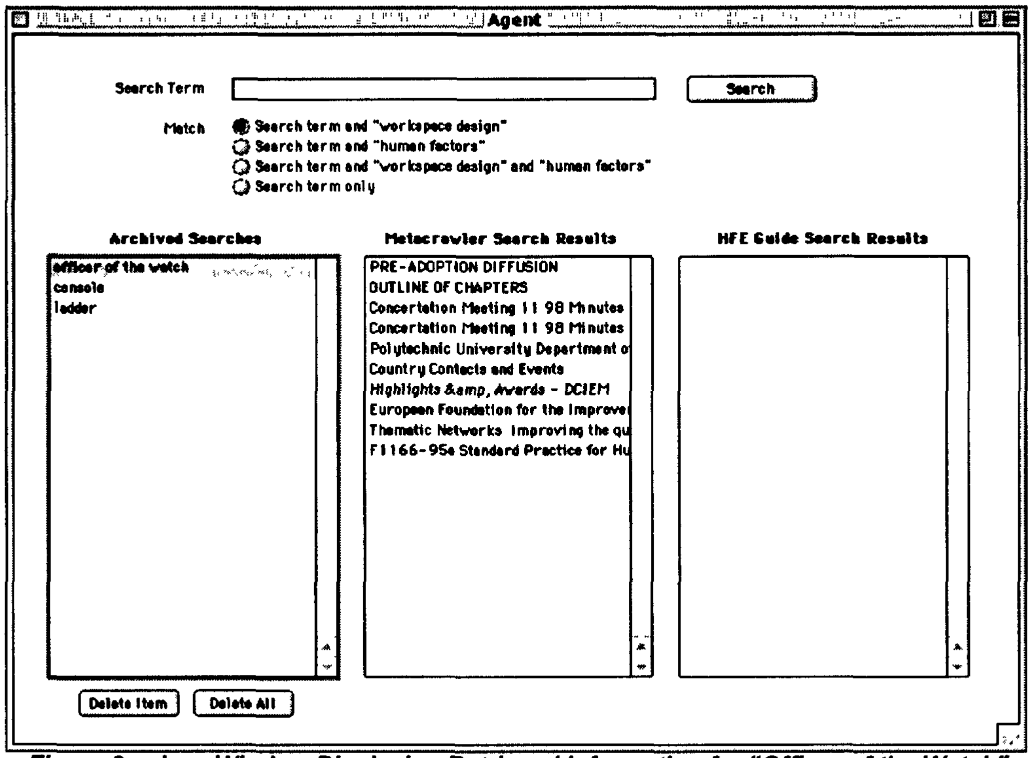


Figure 1b. Deletion of Workstation Lightbulb After Selection

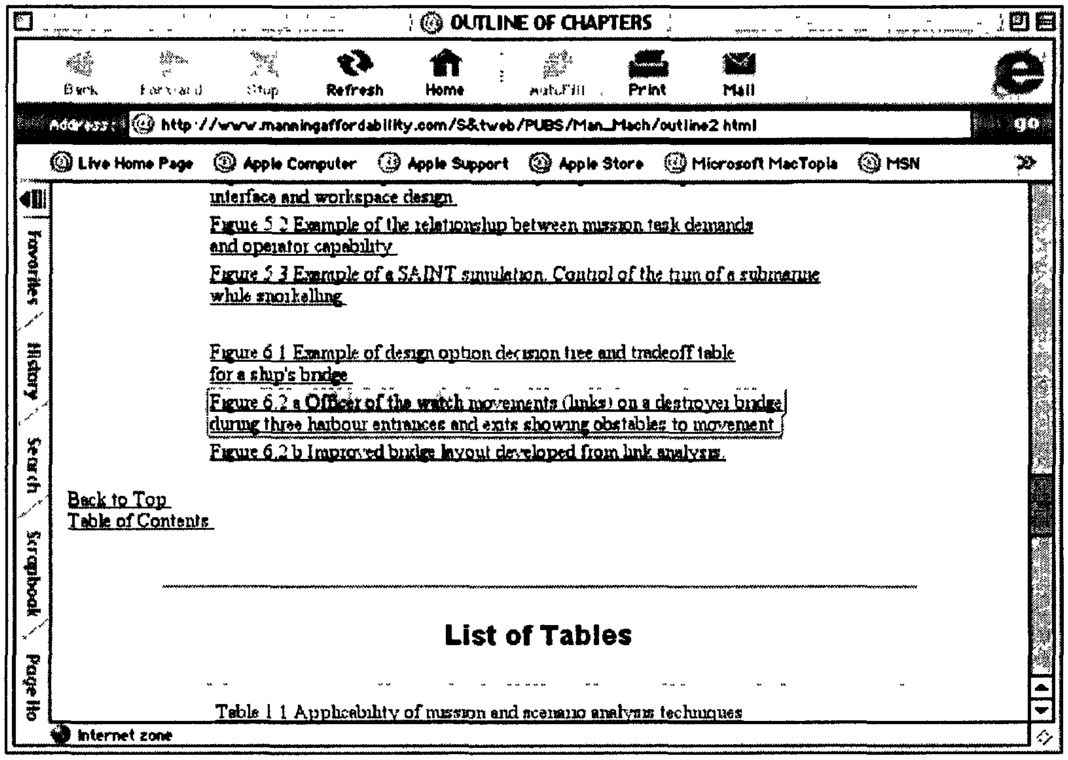


[✓ Search term and "workspace design"]
Search term and "human factors"
Search term and "workspace design" and "human factors"
Search term only

Archived Searches:
officer of the watch
console
ladder

Metacrawler Search Results:
PRE-ADOPTION DIFFUSION
OUTLINE OF CHAPTERS
Concertation Meeting 11 98 Minutes
Concertation Meeting 11 98 Minutes
Polytechnic University Department o
Country Contacts and Events
Highlights &amp; Awards - DCIEM
European Foundation for the Improve
Thematic Networks Improving the qu
F1166-95a Standard Practice for Hu

Figure 2a. Java Window Displaying Retrieved Information for "Officer of the Watch"



http://www.manningaffordability.com/S&tweb/PUBS/Man_Mach/outline2.html

interface and workspace design
Figure 5.2 Example of the relationship between mission task demands and operator capability
Figure 5.3 Example of a SAINT simulation. Control of the trim of a submarine while snorkelling

Figure 6.1 Example of design option decision tree and tradeoff table for a ship's bridge
Figure 6.2 a Officer of the watch movements (links) on a destroyer bridge during three harbour entrances and exits showing obstables to movement
Figure 6.2 b Improved bridge layout developed from link analysis

Back to Top
Table of Contents

List of Tables

Table 1.1 Applicability of mission and scenario analysis techniques

Figure 2b. HTML Browser Window Displaying Information for "Officer of the Watch"

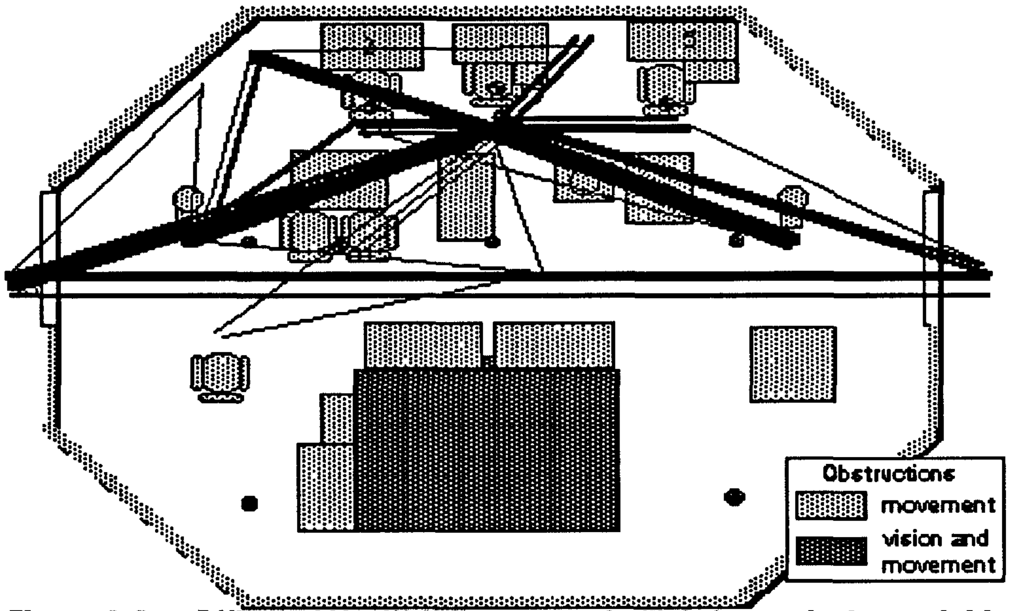
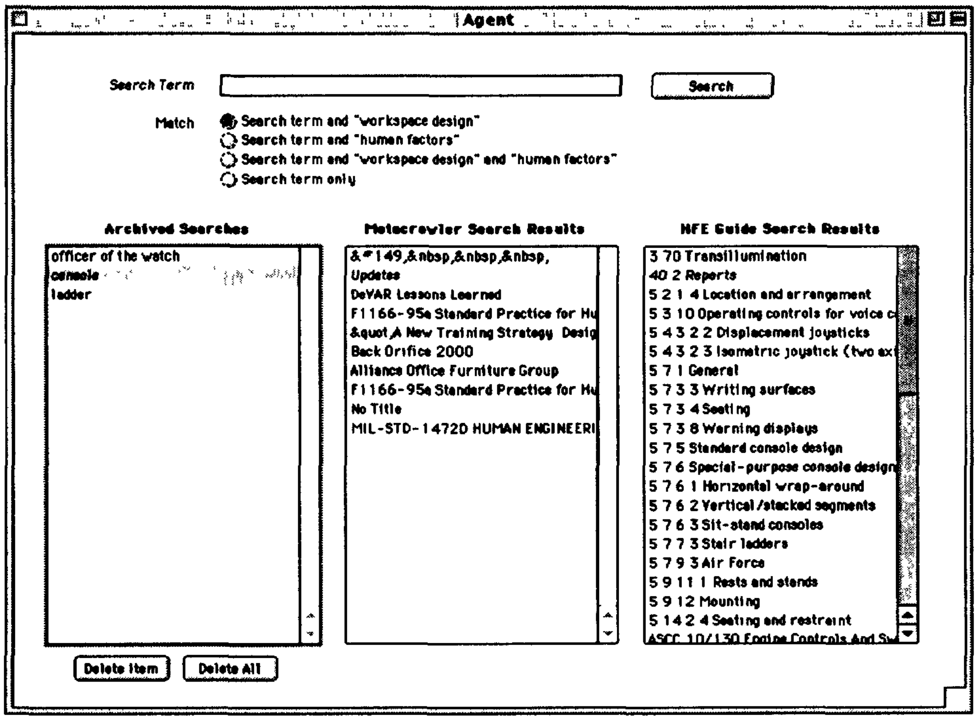


Figure 6.2.a: Officer of the Watch movements (links) on a destroyer bridge during three harbour entrances and exits showing obstacles to movement

Figure 2c. Hyperlinked Figure Illustrating Officer of the Watch Movements(from Agent Results)



Search Term

Match
- [✓ Search term and "workspace design"]
- Search term and "human factors"
- Search term and "workspace design" and "human factors"
- Search term only

Archived Searches

officer of the watch
console
ladder

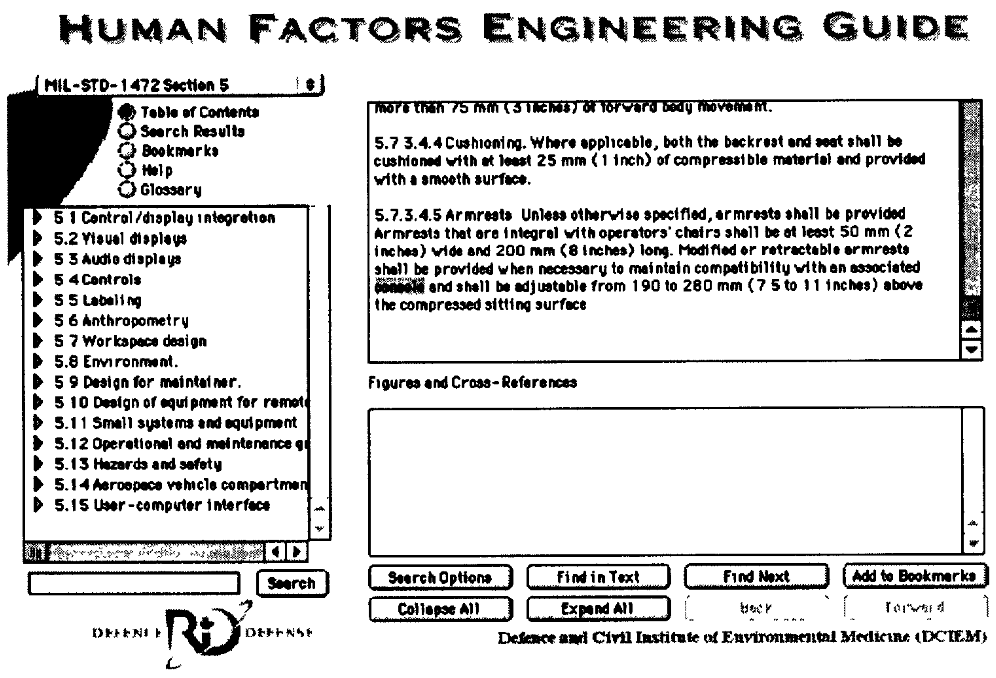
Metacrawler Search Results

&#149, , , ,
Updates
DeVAR Lessons Learned
F1166-95a Standard Practice for Hu
&quot,A New Training Strategy Desig
Back Orifice 2000
Alliance Office Furniture Group
F1166-95a Standard Practice for Hu
No Title
MIL-STD-1472D HUMAN ENGINEERI

HFE Guide Search Results

3 70 Transillumination
40 2 Reports
5 2 1 4 Location and arrangement
5 3 10 Operating controls for voice c
5 4 3 2 2 Displacement joysticks
5 4 3 2 3 Isometric joystick (two ex
5 7 1 General
5 7 3 3 Writing surfaces
5 7 3 4 Seating
5 7 3 8 Warning displays
5 7 5 Standard console design
5 7 6 Special-purpose console design
5 7 6 1 Horizontal wrap-around
5 7 6 2 Vertical/stacked segments
5 7 6 3 Sit-stand consoles
5 7 7 3 Stair ladders
5 7 9 3 Air Force
5 9 11 1 Rests and stands
5 9 12 Mounting
5 14 2 4 Seating and restraint
ASCC 10/130 Engine Controls And Sw

Figure 3a. Java Window Displaying Retrieved Information for "Console"


HUMAN FACTORS ENGINEERING GUIDE
MIL-STD-1472 Section 5
Table of Contents
Search Results
Bookmarks
Help
Glossary
5 1 Control/display integration
5.2 Visual displays
5 3 Audio displays
5 4 Controls
5 5 Labeling
5 6 Anthropometry
5 7 Workspace design
5.8 Environment.
5 9 Design for maintainer.
5 10 Design of equipment for remot
5.11 Small systems and equipment
5.12 Operational and maintenance g
5.13 Hazards and safety
5.14 Aerospace vehicle compartmen
5.15 User-computer interface
Search
Search Options
Find in Text
Find Next
Add to Bookmarks
Collapse All
Expand All
Back
Forward
Defence and Civil Institute of Environmental Medicine (DCIEM)

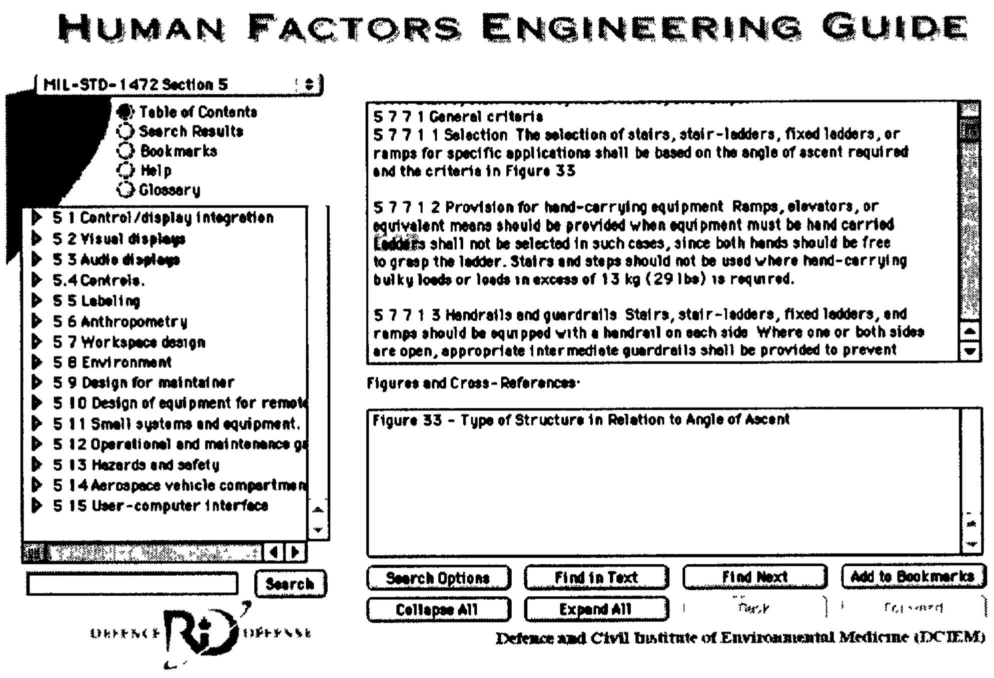
more than 75 mm (3 inches) of forward body movement.

5.7 3.4.4 Cushioning. Where applicable, both the backrest and seat shall be cushioned with at least 25 mm (1 inch) of compressible material and provided with a smooth surface.

5.7.3.4.5 Armrests Unless otherwise specified, armrests shall be provided Armrests that are integral with operators' chairs shall be at least 50 mm (2 inches) wide and 200 mm (8 inches) long. Modified or retractable armrests shall be provided when necessary to maintain compatibility with an associated console and shall be adjustable from 190 to 280 mm (7 5 to 11 inches) above the compressed sitting surface

Figures and Cross-References

Figure 3b. HFE Guide Window Displaying Information for "Console"


HUMAN FACTORS ENGINEERING GUIDE
MIL-STD-1472 Section 5
Table of Contents
Search Results
Bookmarks
Help
Glossary
5 1 Control/display integration
5 2 Visual displays
5 3 Audio displays
5.4 Controls.
5 5 Labeling
5 6 Anthropometry
5 7 Workspace design
5 8 Environment
5 9 Design for maintainer
5 10 Design of equipment for remot
5 11 Small systems and equipment.
5 12 Operational and maintenance g
5 13 Hazards and safety
5 14 Aerospace vehicle compartmen
5 15 User-computer interface
Search
Search Options
Find in Text
Find Next
Add to Bookmarks
Collapse All
Expand All
Back
Forward
Defence and Civil Institute of Environmental Medicine (DCIEM)

5 7 7 1 General criteria
5 7 7 1 1 Selection The selection of stairs, stair-ladders, fixed ladders, or ramps for specific applications shall be based on the angle of ascent required and the criteria in Figure 33

5 7 7 1 2 Provision for hand-carrying equipment Ramps, elevators, or equivalent means should be provided when equipment must be hand carried Ladders shall not be selected in such cases, since both hands should be free to grasp the ladder. Stairs and steps should not be used where hand-carrying bulky loads or loads in excess of 13 kg (29 lbs) is required.

5 7 7 1 3 Handrails and guardrails Stairs, stair-ladders, fixed ladders, and ramps should be equipped with a handrail on each side Where one or both sides are open, appropriate intermediate guardrails shall be provided to prevent

Figures and Cross-References·

Figure 33 - Type of Structure in Relation to Angle of Ascent

Figure 4. HFE Guide Window Displaying Information for "Ladder"

Easing the Data Entry Burden for *LOCATE* Users

A key goal of *LOCATE's* agent network is to provide information that will ease the data entry burden for its users. The focus of the current work has been to lay a foundation that can help achieve that goal.

Building that foundation consisted of selecting a context for constructing an agent network, identifying information retrieval as the key activity for the network, designing and implementing a set of agents to support that and related activities, creating an interface for the presentation of information retrieved and co-ordinating agent network activities with those of *LOCATE* for motivating search and retrieval and notifying users that information is available.

Future work will involve targeting human factors and other sites (besides the HFE Guide site targeted as part of the current contract) that have information helpful to *LOCATE* users, searching those sites periodically to uncover new information and making decisions about specific information on those sites that could be useful in the context of a *LOCATE* user's current activities.

Those efforts should be combined with the addition of information within *LOCATE* itself about how data entry decisions are best made. Available *LOCATE* functions need to be clearly identified and examples of their use provided. The user should understand what those functions are, when it is appropriate to use them and how to decide on their argument values Further, users need help in making those decisions in the context of the design tasks they are performing.

As part of the current contract, a modest beginning was made to providing users, especially new users, with an understanding of the functions and arguments necessary for effective data entry. Information on the nature of the various functions was obtained from Hendy's (1984) thesis and incorporated into *LOCATE's* Internet help files. That information describes the functions in general terms and provides rudimentary help with data entry. Information related to those functions, derived from searches of many other Internet sites, will be a first step in future work on the agent network.

Ultimately, the acquired information should be refined and stored within *LOCATE's* internal help system or its modified Internet help files. Additional information, from searches of various human factors sites, would then be used to supplement that information, providing more in-depth guidance as to which functions a user should employ and what argument values are appropriate to his or her current design task.

A annotated list of source code files, including the new work on agents, appears in Annex A A similar list is provided in Annex B for the standalone Manning Agent Network, which is described the next section.

Generalising *LOCATE's* Agent Technology to Aid Search in Naval Manning Reduction

The purpose of this contract was to investigate and demonstrate the feasibility of world-wide-web based software agents for use in human factors engineering studies.

Subsequent to initiating the work on the design and construction of an agent technology for *LOCATE*, the Defence Research and Development Canada (DRDC) initiated a high-profile study of Technologies for Reducing Crewing aboard Canadian navy ships. That work involves a continuing review of information about technologies for reducing ships crews. The time and resources available for the study are constrained.

The software "agents" developed in the first phase of this contract work provided the ability to scan the world-wide-web, theoretically, on a continuing basis using very limited resources. As such, it is a worked example of the application of agent technology for general human factors studies

An extension to that work was intended as a generalisation of the *LOCATE* agent network to support information retrieval in the aforementioned content area of reducing crewing aboard Canadian navy ships. Beyond that specific purpose, if the generalisation is shown to be useful for identifying sources of information about crewing, then it should be applicable to other human factors integration domains of Personnel, Training, Systems Safety and Health Hazards, as well as Human Factors Engineering.

Developing a Manning Agent Network

Generalising *LOCATE's* Agent Network, the following set of agents were identified and implemented to search for and retrieve publications and web sites related to manning affordability:

- Information Needs Agent (INA);
- Front-end Filtering Agent (FFA);
- Information Retrieval Agent (IRA);
- Back-end Filtering Agent (BFA),
- Information Display Agent (IDA);

The INA identifies informational needs from search terms entered into a window that is accessible from the agent network's main results window. That window is invoked from the main window by clicking on a button that reads, "Modify Search." The search terms are pre-specified but may be changed by the user. Examples of the two windows appear below in Figures 5 and 6 with the initial content of the search window shown.

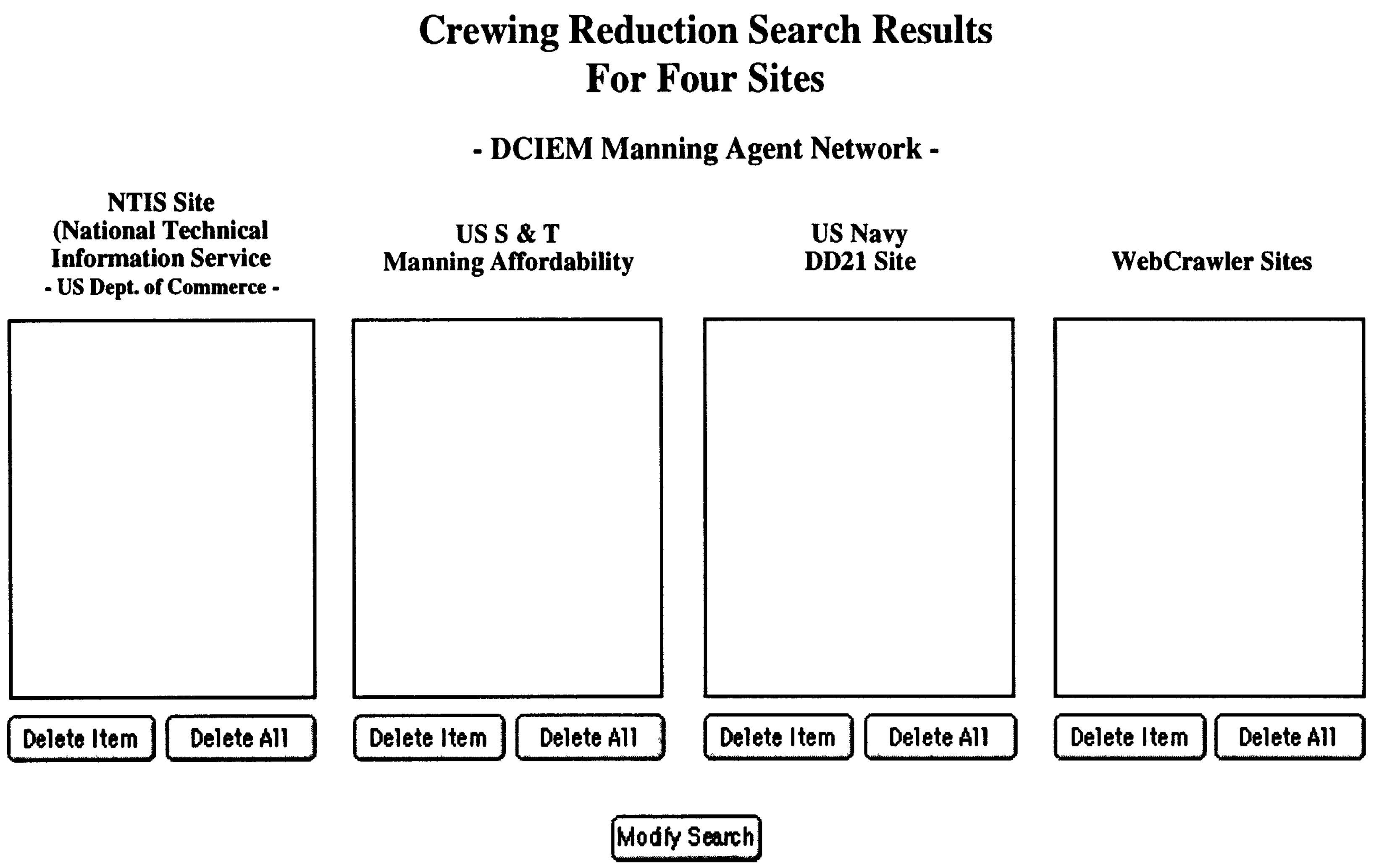
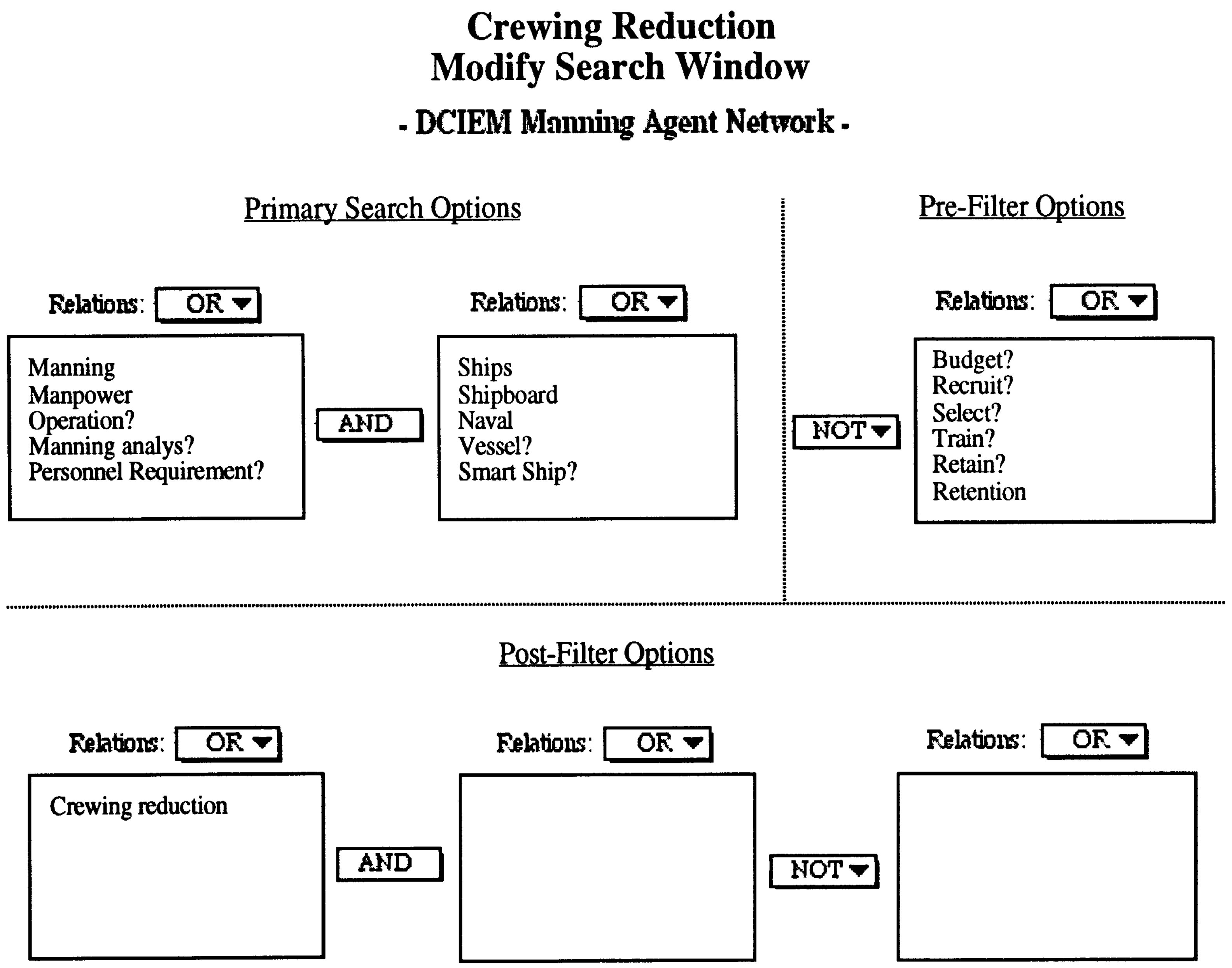


Figure 5. Main Results Window for Manning Agent Network



Crewing Reduction
Modify Search Window

- DCIEM Manning Agent Network -

Primary Search Options

Relations: OR ▼

Manning
Manpower
Operation?
Manning analys?
Personnel Requirement?

AND

Relations: OR ▼

Ships
Shipboard
Naval
Vessel?
Smart Ship?

Pre-Filter Options

NOT ▼

Relations: OR ▼

Budget?
Recruit?
Select?
Train?
Retain?
Retention

Post-Filter Options

Relations: OR ▼

Crewing reduction

AND

Relations: OR ▼

NOT ▼

Relations: OR ▼

Figure 6. Modify Search Window for Manning Agent Network

The panels in Figure 5 are for the display of results from four different sources. Three focused searches are done with the first being the US Commerce's National Technical Information Service (NTIS) database, which contains over 2 million records. The second targeted search site is of the US Science and Technology's (S&T's) Manning Affordability site and the third is the US Navy's DD21 site. The final search is a general search using MetaCrawler, which itself uses a variety of Internet search engines in conducting a search. The addresses of the three targeted sites are as follows:

- NTIS: http://www.ntis.gov/
- Manning Affordability: http://www.manningaffordability.com/s&tweb/index_main.htm
- Navy DD21: http://sc21.crane.navy.mil/

In Figure 6, the Modify Search Window contains a default set of search terms, generated in consultation with the Scientific Authority and organised according to a set AND/OR relations. It also contains a panel for specifying NOT relations, which conceptually is considered part of a front-end filtering process (Agent).

Both words and phrases may be entered into any of the panels. The specification of phrases is done by simply typing each phrase on a separate line: "personnel requirement" is an example as is "crewing reduction" in the bottom section of the window in the figure.

The INA passes the initial search terms to a Font-end Filtering Agent (FFA) which adds the NOT relations and provides any other pre-filtering deemed necessary. That agent then hands off the refined set of search terms and any other qualifications[3] to the Information Retrieval Agent (IRA).

The IRA uses those terms to conduct a search of the Internet and return a set of results from each of the four sites it searches.

As the results are returned, they are handed off to a Back-end Filtering Agent (BFA), which examines its results database to determine if any new results have been returned. If new results are found, the BFA navigates to each new site to determine whether it is valid or not, that is, whether it contains, "Page Not Found," File Not Found," 404 Error," etc., before subjecting the page to further search.

More specifically, the BFA performs the following functions:

- maintains a database of previously found links (both valid and invalid);
- investigates whether a link is valid, that is, whether clicking on it takes the user to a valid site a "Page Not Found," "File Not Found", "404 Error" page, etc..
- adds tags to links indicating whether they are valid or invalid;
- compares new results with old to determine if new sites have been found;
- investigates new links to determine validity;
- adds new links (both valid and invalid) to its database;

[3] In the current implementation, only NOT relations are handled by this agent

The reference to "new" items, links or results, is based on the fact that the Manning Agent Network conducts its searches periodically, in the present case, once an hour, as long as the network application is open. A key task of the BFA, before checking the validity of links, is to determine whether the results of the IRA match a list of previously found items. For those items not part of the list, the agent checks their validity before passing them on for further search and display.

Once it has been determined that a site is valid, the Internet address is passed back to the IR Agent, along with the search terms in the Post-Filtering panels at the bottom of Figure 6. The current example shows only "crewing reduction" as the search phrase used. Further search is then made of the site, which may consist of web pages or abstracts of articles. In future, it appears that it will be possible to conduct searches of PDF documents that reside on a site.

Once sites have been searched and further filtering done, the results are passed to an Information Display Agent (IDA), which decides how the information is to be displayed to the user. In the current implementation the form of that display is fixed but in future versions of the Manning Agent Network, decisions about when and how to display the retrieved information will involve more complex and adaptive decision making.

The filtered results are displayed in their respective panels in the Main Results Window (see Figure 5, above) and the user can then click on any of the items in those panels to view the selected result in an HTML window of an Internet Browser. Depending on the link of course, clicking on an item takes the user to a web page or starts a download of some relevant abstract or article.

Future Work on *LOCATE* and Manning Agent Networks

The *LOCATE* Network

Considerable additional work is possible on *LOCATE's* Agent Network. Since the principal reason for developing such a network is to facilitate the job of *LOCATE* users to make effective decisions about what data should to be entered that best describe their particular designs, the *LOCATE* Agent Network could be used to help retrieve and refine information on the Internet that will help users in that task.

The retrieved information should be studied by *LOCATE* developers, summarised, refined and incorporated into *LOCATE's* intelligent aiding system. In that way, *LOCATE* more effectively can aid users with learning and using its data entry facilities. Subsequent information retrieved by the *LOCATE* Agent Network then will serve to elaborate the knowledge a user possesses as to the data entry options for the user's design.

Co-ordination between *LOCATE's* intelligent aiding system and the Agent Network can be extended by incorporating a knowledge of the information provided to users by the Agent Network into *LOCATE's* beliefs about what users know. Inferences then could be drawn about such things as a user's competency in modelling communication among entities in a design and his or her understanding of *LOCATE* facilities that support that process and how they are to be used.

In the long view, *LOCATE's* network of agents could be extended to support communication with agents in other applications, e.g., those in the suite of tools of which *LOCATE* is a part.[4]

The Manning Network

With respect to the Manning Agent Network, more effective search and retrieval strategies could incorporate statistical procedures similar to those used in some search engines and, ones that could be refined to support more effective search on targeted sites By exploring a variety of search terms, it should be possible to add, subtract and modify the current default terms so that more relevant information on manning is retrieved. Abstracts, articles and web pages found to be particularly useful could be analysed for their content. Effective analytical techniques should lead to the identification of search terms, phrases and concepts having a higher probability of returning relevant material on manning and a better success rate at eliminating material that is not relevant

Filtering could be extending in a variety of ways. Some additions to the pre-filtering NOT category is an extension much like that just mentioned. As the Manning Agent Network's database of sites grows, there should be an exponential decline in new information retrieved. Other, more focused pre-filtering techniques should emerge as the amount of information achieves stability.

[4] That suite of tools is composed of: 1) *LOCATE*, 2) SAFEWORK, a commercially available, computer-based three-dimensional man-model based on anthropometric data from US and Canadian surveys; and, 3) IPME (Integrated Performance Modelling Environment) a tool for conducting front-end human engineering and human performance modelling for validation and verification analysis.

Regarding the Back-end Filtering Agent (BGA) , a number of possibilities exist for extending its functionality. Among those are:

- investigate whether a site includes links to other sites;
- search for and investigate those links;
- notify Web administrators when a link appears to be invalid, providing a return email address that would be monitored by a human in helping to determine the true status of any questionable links;
- periodically re-investigate sites using same search terms;
- note whether new searches have found the same invalid links;
- periodically re-visit invalid links to determine if they still invalid;
- send follow-up email to web administrators regarding invalid links,
- following a pre-determined number of searches, if a previously identified invalid link is not found, tag the link as permanently invalid and not subject to future search.

With respect to the Information Retrieval Agent, searches of web pages could be expanded to include PDF files and other possibly searchable file types. That might involve downloading, converting, searching and discarding files such as postscript files.

Finally, other agents might be added to the Network, e.g., a Site Notification Agent, whose job it would be to detect the presence of facilities for alerting the agent when new additions are made to a site and then to such alerts trigger a new search of the site. Such an approach would mean that searches would be motivated not simply by the passage of time or user activity but by messages from other sites. Those alerts should come from sites that previously had been found to contain content of particular relevance to manning issues. The NTIS site is a site that provides such a facility.

References

Broadbent, G. (1988). *Design in architecture.* London: David Fulton Publishers.

Edwards, J. L., & Hendy, K.C. (2000). A Testbed for Intelligent Aiding in Adaptive Interfaces. Paper presented at the Spring Symposium on Adaptive User Interfaces. Stanford University, March 20-22, 2000. Technical Report SS-00-01, AAAI Press.

Gruber, T.R. (1995). Towards Principles for the Design of Ontologies used for Knowledge Sharing, *International Journal of Human-Computer Studies, 43(5/6),* 907-928.

Hendy, K. C. (1984). 'Locate': A program for computer-aided workspace layout. Master's Thesis, Department of Electrical Engineering, Monash University, Clayton, Victoria, Australia.

Hendy, K. C. (1989). A Model for Human-Machine-Human Interaction in Workspace Layout Problems. *Human Factors, 31(5),* 593-610.

Simon, H. (1981). *The sciences of the artificial.* 2nd edition. Cambridge, MA: MIT Press.

Weiss, G. (Ed.) (1999). *Multiagent Systems : A Modern Approach to Distributed Artificial Intelligence.* New York: MIT Press.

Wooldridge, M. (2000). *Reasoning about Rational Agents.* New York· The MIT Press

Annex A

Annotated List of *LOCATE* Source Code Files

Artificial Intelligence
Management and Development Corporation

Notes:

i) .c files are C source code
.h files are C header files
.rc files are Open Interface resource description files

ii) Some extraneous files in the *LOCATE* folders are outdated and will be erased in future.

AEVENT.C

- This file contains the code to handle Apple Events for opening documents by double-clicking in the Finder.

ALLOBJINFO.C, ALLOBJINFO.RC

- The "AllObjInfo" module contains the code and Open Interface resources necessary for the "All Objects Info" window.

AOBS.C, AOBS.RC

- The "AObs" module contains the code and Open Interface resources necessary for the "Fixed Obstruction" window.

ASSIGN.C

- Original *LOCATE* C file

CFALERT.C, CFALERT.RC

- The "CFAlert" module contains the code and Open Interface resources necessary for the alert box that appears when there have been changes to the design but no cost function has been run since those changes have been made. The alert box appears before displaying the Cost Function History window .

CFBROWSE.C, CFBROWSE.RC

- The "CFBrowse" module contains the code and Open Interface resources necessary for the "Cost Function History" window.

CFCHECK.C, CFCHECK.RC

- The "CFCheck" module contains the code and Open Interface resources necessary for the "Cost Function Checks" window.

COSTCOLR.C, COSTCOLR.H, COSTCOLR.RC

- The "CostColr" module contains the code and Open Interface resources necessary for the "Cost Display Editor" window.

COSTDISP.C, COSTDISP.H, COSTDISP.RC

- The "CostDisp" module contains the code and Open Interface resources necessary for the "Cost Display" window.

COSTFN.C, COSTFN.RC

- The "CostFn" module contains the code and Open Interface resources necessary for the "Cost Function" window.

DRAWROTD.C, DRAWROTD.H

- C code for handling the drawing of rotated objects

DXF.H

- Header file with DXF format constants

DXFOPT.C, DXFOPT.H

- The "DXFOpt" module contains the code and Open Interface resources necessary for the "DXF Import Options" window.

EDITOR.C, EDITOR2.C, EDITOR.RC

- The "Editor" module contains the code and Open Interface resources necessary for the main *LOCATE* window (includes code for Diagrammer, palette, rulers).

EVAL1.C

- Original *LOCATE* C file

EWATTR.C, EWATTR.H, EWATTR.RC

- The "EWAttr" module contains the code and Open Interface resources necessary for the "Workstation" window, which now includes separate tabbed sections for Attributes, Link Functions and Priority Weights.

EXTERN.H

- Original *LOCATE* header file

FORMAT.H

- Original *LOCATE* header file

FUNCT1.C

- Original *LOCATE* C file

GOALOBJ.CPP

- Contains code for defining and handling the C++ goal object

HEADER.DXF

- Contains information that gets added to all exported DXF files

HELPALRT.C, HELPALRT.RC

- The "HelpAlrt" module contains the code and Open Interface resources necessary for displaying the "Help Message" windows.

HELPMORE.C, HELPMORE.RC

- The "HelpMore" module contains the code and Open Interface resources necessary for displaying the "Help Message" windows containing the "More" button.

IMPRTDXF.C

- C code for handling the importing of a workspace from DXF format

INFOUPD.C, INFOUPD.H, INFOUPD.RC

- The "InfoUpd" module contains the code and Open Interface resources necessary for the "Information Update" window.

INFOWIN.C

- C code for handling the "Object Info" window

LFSUMM.C, LFSUMM.RC

- The "LFSumm" module contains the code and Open Interface resources necessary for the "Link Function Summary" window.

LINKDISP.C, LINKDISP.RC

- The "LinkDisp" module contains the code and Open Interface resources necessary for the "Link Display" window.

***LOCATE*.C**

- Based on the original *LOCATE*.C file, this contains the code necessary for loading in a workspace and for computing the cost function.

***LOCATE*.DAT**

- Open Interface compiled resources that are used by *LOCATE* at run-time

LOCATE

- The *LOCATE* application

***LOCATE*.H**

- Original *LOCATE* header file

***LOCATE*.µ**

- *LOCATE* project for CodeWarrior 11

***LOCATE*.RC**

- Open Interface resources in text format

LOCNEW.C

- C code for handling the creation of a new workspace

LOCNEWEW.C

- C code for handling the creation and deletion of workstations and obstructions

LOCSAVE.C

- C code for handling the saving of a workspace

LOCSAVEDXF.C

- C code for handling the saving of a workspace in DXF format

MAIN.C, MAIN.RC

- The "Main" module contains the "main" function which starts up the application.

MISC.C

- Original *LOCATE* C file

MISCRSRC.RC

- The "MiscRsrc" module contains Open Interface resources needed by the application (primarily menu and icon resources).

MULTIOBJ.C, MULTIOBJ.RC

- The "MultiObj" module contains the code and Open Interface resources necessary for the "Multiple Object Creation" window.

NEWUSER.C, NEWUSER.H, NEWUSER.RC

- The "NewUser" module contains the code and Open Interface resources necessary for the "About You" window.

OPT.C

- Optimiser code for changing positions and angles

OPTIM.C

- Original *LOCATE* C file

OPTOPT.C, OPTOPT.RC

- The "OptOpt" module contains the code and Open Interface resources necessary for the "Optimiser Options" window.

OPTSET.C, OPTSET.RC

- The "OptSet" module contains the code and Open Interface resources necessary for the "Optimiser Settings" window.

OPTSTAT.C, OPTSTAT.RC

- The "OptStat" module contains the code and Open Interface resources necessary for the "Optimiser Status" window.

OPTSWAP.C

- Optimiser code for swapping workstations

ORIGIN.C

- Original *LOCATE* C file

OTHEROBJ.C, OTHEROBJ.RC

- The "OtherObj" module contains the code and Open Interface resources necessary for the "Other Object" window.

OUTPUT.C

- Original *LOCATE* C file

PALEDIT.C, PALEDIT.RC

- The "PalEdit" module contains the code and Open Interface resources necessary for the "Palette Editor" window.

PLANS.C, PLANS.RC

- The "Plans" module contains the code and Open Interface resources necessary for the "Plans" window.

PLANREC.C

- The "PlanRec" module contains the code that creates the plan fragments and performs plan recognition.

PRINTPREV.C, PRINTPREV.RC

- The "PrintPrev" module contains the code and Open Interface resources necessary for the "Print Preview" window.

RULEEW.C, RULEEW.H, RULEEW.RC

- The "RuleEW" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Workstation attributes.

RULEGEN.C, RULEGEN.H, RULEGEN.RC

- The "RuleGen" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up object attributes.

RULEOB.C, RULEOB.H, RULEOB.RC

- The "RuleOB" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Obstruction attributes.

RULEOO.C, RULEOO.H, RULEOO.RC

- The "RuleOO" module contains the code and Open Interface resources necessary for the window that informs the user about double-clicking to bring up Other Object attributes

RULER.C, RULER.RC

- The "Ruler" module contains the code and Open Interface resources necessary for the "Ruler" window.

SMRTHELP.C, SMRTHELP.RC

- The "SmrtHelp" module contains the code and Open Interface resources necessary for the "Smart Help" window.

SPLASH.C, SPLASH.RC

- The "Splash" module contains the code and Open Interface resources necessary for the startup screen.

SPLASH2.C, SPLASH2.H, SPLASH2.RC

- The "Splash2" module contains the code and Open Interface resources necessary for the "More on *LOCATE*" window.

START.C, START.RC

- The "Start" module contains the code and Open Interface resources necessary for the usability "Start" window

STARTUP.C, STARTUP.RC

- The "Startup" module contains the code and Open Interface resources necessary for the help reminder at startup.

STARTLOG.C, STARTLOG.RC

- The "StartLog" module contains the code and Open Interface resources necessary for the startup window that allows the user to enter a user name for the help system.

SYSMODL.C, SYSMODL.RC

- The "SysModl" module contains the code and Open Interface resources necessary for the "System Model" window.

TASKMODL.C, TASKMODL.RC

- The "TaskModl" module contains the code and Open Interface resources necessary for the "Task Model" window.

TREE.C

- The "Tree" module contains the code that constructs and searches the plan hierarchy.

USERMODL.C, USERMODL.RC

- The "UserModl" module contains the code and Open Interface resources necessary for the "User Model" window.

WEBBROWS.C, WEBBROWS.H, WEBBROWS.RC

- The "WebBrows" module contains the code and Open Interface resources necessary for the "Web Browser" window.

WOBS.C, WOBS.RC

- The "WObs" module contains the code and Open Interface resources necessary for the "Elemental Obstruction" window.

WSATTR.C, WSATTR.RC

- The "WSAttr" module contains the code and Open Interface resources necessary for the "Workspace Attributes" window.

Annex B

Annotated List of Manning Agent Source Code Files

AI→M

Artificial Intelligence
Management and Development Corporation

AGENTAPPCLASS.JAVA

- This file contains the Java code that initializes the Manning Agent.

BG1.JPG

- This graphic file contains the background image for the "Search Results" window.

BG2.JPG

- This graphic file contains the background image for the "Modify Search" window.

BROWSERLAUNCHER.JAVA

- This file contains the Java code that handles launching the default web browser and displaying a specified URL.

HELP.JPG

- This graphic file contains the help icon.

HELP1.TXT

- This file contains the text that appears in the help window that is accessed from the "Search Results" window.

HELP2.TXT

- This file contains the text that appears in the help window that is accessed from the "Modify Search" window.

HELPFRAME.JAVA

- This file contains the Java code that handles the two help windows.

JAVA.ZIP [PC ONLY]

- This file contains the Java archive needed to run the Manning Agent application on the PC.

MAGENT.MCP

- This is the Metrowerks CodeWarrior project file for the Manning Agent.

MANNING AGENT

- This is the Manning Agent application.

MODIFYFRAME.JAVA

- This file contains the Java code that relates to the "Modify Search" window.

RESULTSFRAME.JAVA

- This file contains the Java code that relates to the "Search Results" window.

SEARCHES.JAVA

- This file contains the Java code that handles the searching of the four sites.

TABS.JPG

- This graphic file contains the image for the tabs in the “Search Results” window.

DOCUMENT CONTROL DATA SHEET

1a. PERFORMING AGENCY Artificial Intelligence Management and Development Corporation, 206 Keewatin Avenue, Toronto, ON M4P 1Z8 CANADA	2. SECURITY CLASSIFICATION UNCLASSIFIED Unlimited distribution -
1b. PUBLISHING AGENCY DCIEM	
3. TITLE (U) Building an agent technology for users of the LOCATE workspace layout tool	
4. AUTHORS Jack L. Edwards	
5. DATE OF PUBLICATION April 30 , 2001	6. NO. OF PAGES 46
7. DESCRIPTIVE NOTES	
8. SPONSORING/MONITORING/CONTRACTING/TASKING AGENCY Sponsoring Agency: Monitoring Agency: Contracting Agency : DCIEM Tasking Agency:	

9. ORIGINATORS DOCUMENT NO. Contract Report CR 2001-126	10. CONTRACT GRANT AND/OR PROJECT NO. PW&GS W7711-00-7661/001/TOR; 3ad13	11. OTHER DOCUMENT NOS. AC213
12. DOCUMENT RELEASABILITY Unlimited distribution		
13. DOCUMENT ANNOUNCEMENT Unlimited announcement		

14. ABSTRACT

(U) Work developed a tight integration between LOCATE's ability to track user activity and a new LOCATE Agent Network whose ultimate goal is to help ease the data entry burden for LOCATE users. A society of agents was created using the JATLite development environment and KQML communication protocols.

An extension to this contract generalised the LOCATE work to the area of manning reduction on naval vessels. Agents conduct periodic searches of the Internet using the same search and filtering techniques.

Future work on LOCATE will use retrieved information to extend LOCATE's internal, intelligent-aiding support for data entry. Future work on the Manning Agent Network will extend the functionality of its filtering agents and refine its search strategies.

15. KEYWORDS, DESCRIPTORS or IDENTIFIERS

(U) HUMAN ENGINEERING TOOLS; HUMAN MODELLING; WORKSPACE LAYOUT; WORKSPACE DESIGN; FACILITY LAYOUT; COMPUTER-AIDED DESIGN; INTELLIGENT HELP; ADAPTIVE INTERFACES; LOCATE; EXPLICIT USER MODELS

516836

CA020160